BFS (H1)

Based on the codes from the book:
Artificial Intelligence : A Modern Approach
The copyrights of the codes belong to
Ravi Mohan, Peter Norvig, Stuart Russell, Ciaran O'Reilly

```java
package aima.search.uninformed;

import java.util.List;

import aima.search.framework.Metrics;
import aima.search.framework.Problem;
import aima.search.framework.QueueSearch;
import aima.search.framework.Search;
import aima.search.nodestore.FIFONodeStore;

/**
 * Artifitial Intelligence A Modern Approach (2nd Edition): page 73.
 *
 * Breadth-first search.
 */
public class BreadthFirstSearch implements Search {

	private final QueueSearch search;

	public BreadthFirstSearch(QueueSearch search) {
		this.search = search;
	}

	public List search(Problem p) {
		return search.search(p, new FIFONodeStore());
	}

	public Metrics getMetrics() {
		return search.getMetrics();
	}

}
```

interface

QueueSearch extends
NodeExapnader

search()
clearInstrumentation()
getQueueSize()
setQueueSize()
getMaxQueueSize()
getPathCost()
setPathCost()
addExpandedNodesToFringe()

NodeExpander

clearInstrumentation()
expandNode()
getNodesExpanded()
setNodesExpanded()
getSearchMetric()
getMetrics()

```java
public Metrics getMetrics() {
	return metrics;
}
```

```
public List<String> search(Problem problem, NodeStore fringe) {
        clearInstrumentation();
        fringe.add(new Node(problem.getInitialState()));
        setQueueSize(fringe.size());
        while (!(fringe.isEmpty())) {
                Node node = fringe.remove();
                setQueueSize(fringe.size());
                if (problem.isGoalState(node.getState())) {
                        setPathCost(node.getPathCost());
                        return SearchUtils.actionsFromNodes(node.getPathFromRoot());
                }
                addExpandedNodesToFringe(fringe, node, problem);
                setQueueSize(fringe.size());
        }
        return new ArrayList<String>();// Empty List indicates Failure
}


public Metrics getMetrics() {
        return metrics;
}
```